

<https://www.itread01.com/p/1407878.html>

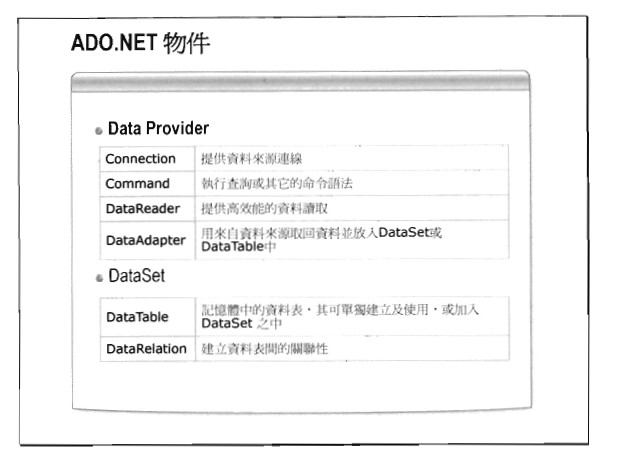
整理以上網址所學和教材互相印證︰( 作者於 2018 年所寫的 )

|  |
| --- |
| 名稱 |
| ADO.NET是ActiveX Data Objects的縮寫,  它是一個COM元件庫,用於在microsoft技術中訪問資料。  之所以叫ADO.NET,應該是微軟自己打的廣告,希望在NET程式設計環境中優先使用這種資料訪問介面。 |

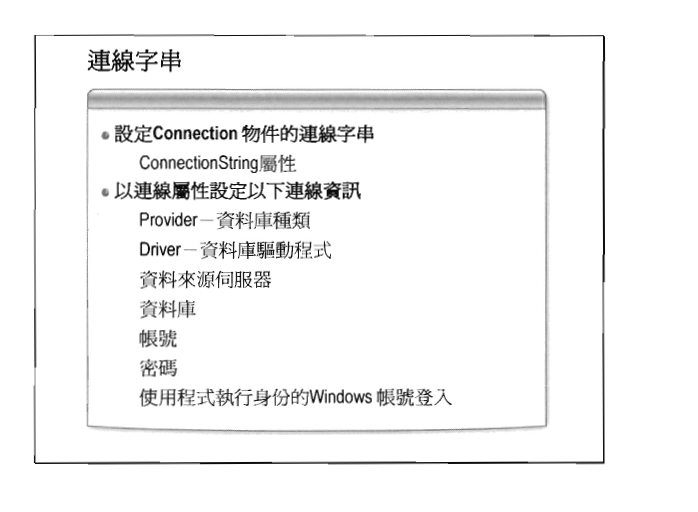
|  |
| --- |
| 功能 |
| 簡單來說,ADO.NET就是一種資料訪問介面,  可以讓我們在程式中呼叫相應的類庫對資料庫  ( 通常為SQL Server , 也可以是access 等其他資料庫 )  進行增刪改查等操作。 |

ADO.NET由五大類庫組成,分別是:

|  |
| --- |
| Connection(用於建立與 資料庫的連線) |
| Command(用於執行SQL語句) |
| DataReader(用於讀取資料) |
| DataAdapter(用於填充把資料填充到DataSet) |
| DataSet(資料集,用於程式中) |



|  |
| --- |
| 引用模組 |
| 要想使用ADO.NET需要在程式中引用 (using) System.Data.SqlClient  引用後，可以操作︰  SqlConnection : 連線資料庫  SqlCommand : 資料庫命名物件  SqlCommandBuilder : 生成SQL命令  SqlDataReader : 資料讀取器  SqlDataAdapter : 資料介面卡,用於填充DataSet  SqlParameter : 為儲存過程定義引數  SqlTransaction : 資料庫事務 |



Driver 分出另外一份 Word 來說。

|  |
| --- |
| Connection |
| 訪問資料庫流程 :  建立一個到資料庫的連線  開啟資料庫連線  …  關閉連線 |
| 連線字串,它的基本語法為 :  Data Source(資料來源) +  Initial Catalog(資料庫名稱) +  User ID(使用者名稱) +  Password(密碼) |
| String connectString = "  Data Source = myServerAddress;  Initial Catalog = myDataBase;  User Id = myUserName; Password = myPassword;";  或  String connectString = "  Server =myServerAddress;  Database = myDataBase;  User Id = myUsername; Password = myPassword;"; |
| 對於Sql Server來說,它支援兩種身份驗證方法,   |  |  | | --- | --- | | Sql Server身份驗證 | windows身份驗證 | | 連結字串還可以包含哪些屬性 ?  要用時再自己查。 | 要用windows身份驗證,  就需要在連線字串中包括Integrated Security屬性，並設定為True(預設False) |   進階的「連線字串生成器」見下方︰ |

|  |
| --- |
| 建立連線物件 |
| SqlConnection connection = new SqlConnection(connecString); |
| 或者用 專門的連線字串生成器 : (這個方式，看起來無比的清淅) |
| SqlConnectionStringBuilder connectionStringBuilder =  new SqlConnectionStringBuilder()  {  DataSource=”“,  InitialCatalog=”“,  UserID=”“,  Password=””  };  SqlConnection connection =  new SqlConnection(connectionStringBuilder.ToString()); |
| 開啟或關閉連線 :  connection.Open();  connection.Close(); |

|  |
| --- |
| Web.config 之中的 <connectionString> 區段 |
|  |
| 如何在主程式讀取<connectionString>設定︰ |
| 前端 ︰ |
| 後端︰  Using System.Web.configuration; |
| 以下是我自己在實務中學到的，更簡潔的方法︰ |
|  |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| SqlCommand命令物件 | | | |
| 它具有四個主要屬性，為︰ | | | |
|  | | 預設值 | 用途 |
| CommandText | | 空字串 | SQL語句 |
| CommandTimeout | | 30 |  |
| CommandType | | CommandType.Text | 有三個列舉 , 分別是  Text(SQL文字命令)  StoredProcedure(儲存過程)  TableDirect(表名) |
| Connection | | Null |  |
|  | | | |
| 建立命令物件: | | | |
| SqlCommand command = connection.CreateCommand(); 或  SqlCommand command = new SqlCommand(); | | | |
| 幾個重要的方法︰ | | | |
| ExecuteNonQuery | 主要執行增刪改操作  回傳一個整數來代表受影響資料筆數 | | |
| ExecuteReader | 返回的是SqlDataReader型別 (一個 DataReader 物件)  ,主要用來查詢 | | |
| ExecuteScalar | 返回執行結果集中的第一行第一列,  如果沒有資料,則返回NULL | | |
| CreateParameter | 下頁說明 | | |
| 例子︰  SqlConnectionStringBuilder conStr = new SqlConnectionStringBuilder();  conSt.DataSource=@".\SQLEXPRESS";  conStr.IntegratedSecurity=true;  conStr.InitialCatalog="db\_Test";  StringBuilder strSQL = new StringBuilder();  for(int i=0;i<=100;i++)  strSQL.Append("insert into table XXXX……. ;");  using(SqlConnection con = new SqlConnection(conStr.ConnectionString))  {  con.Open();  SqlCommand cmd = new SqlCommand(strSQL.ToString() , con );  int impactedNumber = cmd.ExecuteNonQuery(); // 返回受影響的行數  object firstData = cmd.ExecuteScalar(); // 返回執行結果中的第一行第一列  } | | | |

|  |
| --- |
| SqlParameter ( 命令中的參數 ) |
| 該類有幾個重要的屬性 :  ParameterName : 設定引數名  Value : 給引數設定值  Size : 設定引數位元組最大長度  SqlDbType : 引數在SQL中的類別 |
| 和幾個重要的方法 :  AddWithVlue  Add  AddRange |
| 例子 : |
| SqlConnection connection =new SqlConnection("XXXXXXXXX"))  {  SqlCommand cmd = connection.CreateCommand();  cmd.CommandText="XXXXXXXXX";  // 方法一  cmd.Parameters.Add("@name",SqlDbType.NVarChar).Value = "deng";  // 方法二  cmd.Parameters.AddWithValue(@"name","deng");  // 方法三  SqlParameter[] parameters = new SqlParameter[]  {  new SqlParameter("@name",SqlDbType.NvarChar,100){Value="deng"},  new SqlParameter("@XXX",SqlDbType.NvarChar,100){Value="XXXXX"},  ……  };  cmd.Parameters.AddRange(parameters);  }  // 可以通過cmd.Parameters[i].Value設定和讀取數值。( = 有 index) |

|  |  |
| --- | --- |
| 資料讀取 SqlDataReader | |
| SqlConnetion con = new SqlConnection("XXXXXXXXXXX")  {  con.Open();  SqlCommand cmd = con.CreateCommand();  cmd.CommandText="";  SqlDataReader dr = cmd.ExecuteReader( CommandBehavior.CloseConnection )  {  While(dr.Read())  {  string str = dr.GetSqlString(0).ToString();  // do sth … ( 把 str 存起來。 )  }  }  }  它是連線相關的  與資料庫的連線一斷開就無法讀取資料庫中的資料  由此可知，查詢結果並不是放在程式中，而是放在資料庫的服務中。 | |
| 常用方法︰ | |
| GetOrdinal | 可以獲取指定列名的序列號,int name = dr.GetOrdinal(“name”); |
| GetName | 與上面的方法對應,可以通過列號返回列名字 |
| IsDBNull | 判斷當前讀取的資料是否為Null |
| NextResult | 當查詢為批處理查詢時, (batch)  使用這個方法去獲取下一個結果集(一個dataSet,多個datatable)  返回值為Bool,如果存在多個結果集,則為true;否則為false |
| Read | 讀取資料 |
| 常用屬性: | |
| HasRow | 是否有資料 |
| FieldCount | 獲取讀取的列數 |
| IsClosed | 判斷讀取的資料流是否關閉 |
| 這代表可以一次下兩句SQL，找兩個不同的表格，回傳一個DataReader，拆結果兩次？ ( 有空測看看。 ) | |

|  |
| --- |
| SqlTransaction (自我理解為︰讀檔點) |
| 該儲存點之後的操作，如出錯都將會回滾。 |
| SqlConnection con = new SqlConnection(strCon);  con.Open();  SqlTransaction transaction = con.BeginTransaction();  SqlCommand cmd = con.CreateCommand();  cmd.CommandText = "XXXXX"  cmd.Transaction = transaction;  transaction.Save("transaction point");  X X X X X X 動作  如出錯︰ ( 🡨 這個怎麼寫我還不知道 )  transaction.Rollback("transaction point") |

|  |
| --- |
| SqlDataAdapter ( 程式與資料庫之間的中介，接收訊息 ) |
| 資料庫  用戶  DataReader 在 DataAdapter 的後方  從資料庫回來時，Adapter把資料放入DataSet , DataTable。  從用戶到資料庫，也可以操作DataReader 那種逆向把新DataSet 轉為Updata 資料的能力 ( 未測 ) |
| 建構函式   1. SqlDataAdapter( ) 2. SqlDataAdapter( SqlCommand ) 3. SqlDataAdapter( String , SqlConnection ) 4. SqlDataAdapter( String , ConnectionString ) |
| 例︰  DataSet dataSet = new DataSet();  SqlConnection con = new SqlConnection("");  con.Open();  SqlCommand cmd = con.CreateCommand();  cmd.CommandText="select xxx from tb\_xxx";  SqlDataAdapter dataAdapter = new SqlDataAdapter(cmd);  dataAdapter.Fill(dataSet); |
| SqlCommandBuilder |
| 把撈回來的DataSet修改後，  呼叫Update方法，把Dataset的資料，轉化為SQL語句用來更新資料庫。例︰ ( 未測 ) |
| SqlConnection con = new SqlConnection(ConnectionString();  con.Open();  SqlDataAdapter da = new SqlDataAdapter("select xxx from tb\_xx");  DataSet ds =new DataSet();  da.Fill(ds);  SqlCommandBuilder cmdBuilder = new SqlCommandBuilder(da);  DataRow row = ds.Tables[0].NewRow(); // 新加一行  row[0]="a"; // 新行的第一欄 = a  row[1]="b"; // 新行的第二欄 = b  ds.Tables[0].Rows.Add(row); // 正式加入  da.Update(ds); // da 更新為 ds // 是不是這樣就可以改到資料庫？ |
| DataSet |
|  |
|  |
| ( 不太明白為什麼 DataSet要命名，可能是我還沒有處理到多個 DataSet ) |
| // 把客戶資料取回放入 DataSet的 DataTable 中，並把DT命名為 Customer |

|  |
| --- |
| DataView 進行資料篩選 |
|  |
| 和 資料庫 的View相似，但無法當成 DataTable 使用  為什麼要View而不是直接從撈回來時就Where？  應該是要節省和資料庫連線的次數。 |
| 建立方式1︰  DataView dv = newDataView(原始的DataTable)  建立方式2︰  DataView dv =原始的DataTable.DefaultView; |
| 使用方法︰ RowFilter  dv.RowFilter = “column= ‘condition’ ”;  dv.RowFilter = “column Like ‘condition%’ ”;  dv.RowFilter = “column Like ‘condition% and XXX ’ ”;  dv.RowFilter = “column Like ‘condition% or XXX ’ ”; |
| 篩選的結果︰不是刪欄位，而是縮減行數。 |
| PDF 35-38 的例子是用Session來存首次的結果，  (繫結GV顯示)  之後查詢條件都用dv來篩。  Session是跨頁應用的？所以其實用ViewState就行？  可以測測。 |

|  |
| --- |
| ASP.NET 資料存取模型 P 39 |
|  |
| 好處︰減少後端的程式碼，不需要手動撰寫ADO物件 |
| 資料繫結控制項(Data Bound Control)︰ |
| Data Source Contorl : |
| P 41 SqlDataSource    回傳預設是DataSet  GridView和DetailsView有排序和分頁，比較適合顯示這種資訊。 |
| 把DataSourceMode屬性設定為DataReader 表示回傳資不會整批存在應用程式侗服器記憶體？是什麼意思？P42 |
| 這個東東，在上一份教材玩過了，所以略… |

|  |
| --- |
| 自訂資料來源控制項參數 P43 |
|  |
| 配合 SqlDataSource 和 ObjectDataSource 的不固定參數 ( 從輸入而來的叫ControlParameter)  如 CookieParameter, FormParameter, ProfileParameter, QueryStringParameter, SessionParameter﹑還可以自訂，以下是範例︰ |
| 自己開一個模組，繼承Parameter，覆蓋Evaluate的方法︰ |
| 在網頁的前端，註冊引入這個模組︰ |
| 在使用SqlDataSource的時候，引用這個參數︰ |
| 優點︰  在維護的時候，參數容易作一次性的修改。(動態的修改可能嗎？不行吧？)  不會因為SqlDataSource 是前端的控制項就完全沒有後端的彈性。  ( 假設是在多個 SqlDataSource 共用一個或多個參數的情況下 ) |
| 上述內容皆未測，只是轉述PDF。 |
| 練習︰ P 46      value 從什麼地方來？base又是來自？    引入namespace    拉入一個GridView > Choose Data Source > New data source > Database  然後按下Add  生成︰      改 SelectParameter 為︰    結果︰    所以︰  在set Year 的時候，模組從前端拿到了1966 為 value  把value設為base.ViewState[“Year”]  用base.OnParameterChange(); 刷新？  於是當GridView生成時，呼叫了SqlDataSource，SqlDataSource去get的時候  Return 了base.ViewState[“Year”]  結論是︰ 這個是從前端到後端又回到前端的例子。 |

P 54 後，介紹 LiewView 又分了一份 Word.